UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/990,802	11/13/2001	Eitan Farchi	SVL920010003US1	3720

25693 7590 02/14/2006

KENYON & KENYON LLP
RIVERPARK TOWERS, SUITE 600
333 W. SAN CARLOS ST.
SAN JOSE, CA 95110

EXAMINER	
MITCHELL, JASON D	

ART UNIT	PAPER NUMBER
2193	

DATE MAILED: 02/14/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application Number: 09/990,802
Filing Date: November 13, 2001
Appellant(s): FARCHI ET AL.

Frank L. Bernstein
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed 11/15/05 appealing from the Office action mailed 6/16/05.

(1) Real Party in Interest

A statement identifying by name the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(3) Status of Claims

The statement of the status of claims contained in the brief is correct.

(4) Status of Amendments After Final

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

(5) Summary of Claimed Subject Matter

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct.

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

5573387	Chen	9-1997
5778169	Reinhardt	7-1998

Windser "Managing data through naming standards" Software, IEEE, Vol. 7.4, 7-1990.

Claims 1, 6-8, 13-15 and 20-21 are rejected as anticipated by Chen, claims 2-3, 9-10, and 16-17 are rejected as unpatentable over Chen in view of Windser, claims 4, 11, and 18 are rejected as unpatentable over Chen in view of Reinhardt, and claims 5, 12 and 19 are rejected as unpatentable over Chen.

(9) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

The Rejection of all claims made in the final office action is being reproduced herein, including changes made to more explicitly address remarks made by Appellant in the Brief filed 11/15/05.

Claims 1, 6-8, 13-15, and 20-21 are rejected under 35 U.S.C. 102(b) as being anticipated by USPN 5,573,387 to Chen et al. (Chen).

Regarding Claim 1: Chen discloses identifying the computer program for which the persistent code coverage data should be collected (col. 2, lines 47-50 'a software system is partitioned in to basic code entities'); dividing the program source code statements of said computer program into a plurality of code coverage tasks (col. 2 lines 47-50 'partitioned into basic code entities'; col. 3, lines 46-61 'Functions are the basic entities that execute program semantics'), generating a persistent (col. 7, line 64-col. 8, line 2 'generates the C program database 177') unique name for each of the code coverage tasks of said plurality of code coverage tasks (col. 9, lines 1-3 'all entities have the attributes ... kind, ... name' and col. 11, lines 20-21 'two entities match if they have the same name and entity kind'); inserting coverage points into the computer program source

code for each of the code coverage tasks to produce an instrumented program (col. 7, lines 7-9 'adding instrumentation to the code, which results in instrumented C source code'). Note that 'function trace lists 161 and 163 through 165' generated by the instrumented code contain sufficient data to determine which 'entities' or 'code coverage tasks' have been executed and consequently there must be instrumentation 'for' each 'entity' (col. 9, lines 32-57)

Further, Chen discloses compiling and linking the instrumented program into a program executable (col. 7, lines 9-11 'Instrumented C source code is then compiled by a C compiler with the results in instrumented C source code'), identifying a set of test cases from a plurality of test cases to be run for the code coverage data collection purposes (col. 11, lines 66-67 'determine which test units ... need to be re-run'); creating a code coverage database using the code coverage tasks and the identified set of test cases (col. 9, lines 32-35 'generate an entity trace list for each test unit'); running the program executable with a test case from the identified set of test cases (col. 7, lines 30-33 'one execution for each of the N test units') and writing the information about the test case and the coverage points that are executed into an output file (col. 7, lines 40-44 'generates a function trace list'), until all the test cases have been run (col. 7, line 32 'for each of the N test units'); and processing the information contained in the output file into code coverage data and populating the code coverage database with said code coverage data (col. 9, lines 32-35 'the function trace lists ... are then used to generate an entity trace list').

Regarding Claim 6: The rejection of claim 1 is incorporated; further, Chen discloses modifying the computer program to produce a modified version of the computer program source code (col. 10. line 50-53 'after modifications have been made'); identifying a plurality of new, modified, and deleted code coverage tasks in said modified version of the computer program source code (col. 10, lines 63-66 'entity difference list'); generating a persistent unique name (col. 9, lines 1-3 'all entities have the attributes ... kind, ... name' and col. 11, lines 20-21 'two entities match if they have the same name and entity kind') for each of the new and modified code coverage tasks of said plurality of new, modified and deleted code coverage tasks; inserting coverage points into the modified version of the computer program source code for each of the new and modified code coverage tasks to produce an instrumented modified version of the computer program source code (col. 7, lines 7-9 'adding instrumentation'); compiling and linking the instrumented modified version of the computer program source code (col. 7, lines 9-11 'compiled by a C compiler') into a modified program executable, identifying a new set of test cases from a plurality of test cases to be run for the code coverage data collection purposes on the new and modified code coverage tasks (col. 11, lines 66-67 'the entity difference list is used ... to determine which tests units ... need to be re-run'); altering the code coverage database to accommodate new, modified and deleted code coverage tasks (col. 12, lines 41-47 'new entity trace lists must be generated for each test unit') and the new set of test cases, and clearing any code coverage data for the modified code coverage tasks (col. 12, lines 41-47 'new entity traces list must be generated') from said

code coverage database; running the modified program executable with a test case from the identified new set of test cases (col. 11, lines 66-67 're-run') and collecting code coverage data for the new and modified code coverage tasks (col. 7, lines 40-44 'generates a function trace list'), until all the test cases have been run (col. 7, line 32 'for each of the N test units'); and updating the code coverage database with the collected code coverage data (col. 9, lines 32-35 'the function trace lists ... are then used to generate') for the new and modified code coverage tasks; whereby the previously collected code coverage data for the non-affected code coverage tasks is preserved (col. 12, lines 41-47 'new entity trace lists must be generated ... covered by each of the selected test units') from a previous version of the computer program to the modified version of said computer program eliminating the need for running the entire test bucket (col. 12, lines 38-40 'selected test units are re-run in order to test the modified software system').

Regarding Claim 7: The rejection of claim 6 is incorporated; further Chen discloses changing the version indicator (col. 8, lines 59-63 'checksum is used ... to determine whether an entity has been changed' and col. 11, lines 26-27 'the checksum ... is retrieved from the second C program database'). While not explicitly stated, the checksum does identify the version of the entity in that it distinguishes between two different versions of said entity.

Regarding Claim 8: Chen discloses an apparatus for collecting persistent code coverage data for a program, the persistent code coverage data being stored in a code coverage database associated with the program (col. 6, lines 50-52 'the

external storage device ... may be used for the storage of data), the program comprising program source code statements (col. 5, lines 33-52), the apparatus comprising: a computer system having a data storage device (col. 6, lines 50-52 'the external storage device ... may be used for the storage of data') connected thereto, wherein the data storage device stores the code coverage database, and one or more computer programs (col. 6, lines 50-52 'the external storage device ... may be used for the storage of ... computer program code') executed by the computer system for: identifying the computer program for which the persistent code coverage data should be collected (col. 2, lines 47-50 'a software system'); dividing the program source code statements of said computer program into a plurality of code coverage tasks (col. 2 lines 47-50 'partitioned into basic code entities'), generating a persistent unique name for each of the code coverage tasks (col. 9, lines 1-3 'all entities have the attributes ... kind, ... name' and col. 11, lines 20-21 'two entities match if they have the same name and entity kind') of said plurality of code coverage tasks; inserting coverage points into the computer program source code for each of the code coverage tasks to produce an instrumented program (col. 7, lines 7-9 'adding instrumentation'), compiling and linking the instrumented program into a program executable (col. 7, lines 9-11 'compiled by a C compiler'), identifying a set of test cases from a plurality of test cases to be run for the code coverage data collection purposes (col. 11, lines 66-67 'determine which test units ... need to be re-run'); creating a code coverage database using the code coverage tasks and the identified set of test cases (col. 9, lines 32-35 'generate an entity trace list for each test unit'); running

the program executable with a test case from the identified set of test cases (col. 7, lines 30-33 'for each of the N test units') and writing the information about the test case and the coverage points that are executed into an output file (col. 7, lines 40-44 'generates a function trace list'), until all the test cases have been run (col. 7, line 32 'for each of the N test units'); and processing the information contained in the output file into code coverage data and populating the code coverage database with said code coverage data (col. 9, lines 32-35 'the function trace lists ... are then used to generate an entity trace list').

Regarding Claim 13: The rejection of claim 8 is incorporated; further, Chen discloses modifying the computer program to produce a modified version of the computer program source code (col. 10. line 50-53 'after modifications have been made'); identifying a plurality of new, modified, and deleted code coverage tasks (col. 10, lines 63-66 'entity difference list') in said modified version of the computer program source code; generating a persistent unique name (col. 9, lines 1-3 'all entities have the attributes ... kind, ... name' and col. 11, lines 20-21 'two entities match if they have the same name and entity kind') for each of the new and modified code coverage tasks of said plurality of new, modified and deleted code coverage tasks; inserting coverage points (col. 7, lines 7-9 'adding instrumentation') into the modified version of the computer program source code for each of the new and modified code coverage tasks to produce an instrumented modified version of the computer program source code; compiling and linking the instrumented modified version of the computer program source code (col. 7, lines 9-11 'compiled by a C compiler') into a modified program

executable, identifying a new set of test cases (col. 11, lines 66-67 'the entity difference list is used ... to determine which tests units ... need to be re-run') from a plurality of test cases to be run for the code coverage data collection purposes on the new and modified code coverage tasks; altering the code coverage database to accommodate new, modified and deleted code coverage tasks (col. 12, lines 41-47 'new entity trace lists must be generated for each test unit') and the new set of test cases, and clearing any code coverage data for the modified code coverage tasks (col. 12, lines 41-43 'new entity traces list must be generated') from said code coverage database; running the modified program executable with a test case from the identified new set of test cases (col. 11, lines 66-67 're-run') and collecting code coverage data for the new and modified code coverage tasks (col. 7, lines 40-44 'generates a function trace list'), until all the test cases have been run (col. 7, line 32 'for each of the N test units'); and updating the code coverage database with the collected code coverage data (col. 9, lines 32-35 'the function trace lists ... are then used to generate') for the new and modified code coverage tasks; whereby the previously collected code coverage data for the non-affected code coverage tasks is preserved (col. 12, liens 41-47 'new entity trace lists must be generated ... covered by each of the selected test units') from a previous version of the computer program to the modified version of said computer program eliminating the need for running the entire test bucket (col. 11, lines 66-67 'which test units ... need to be re-run').

Regarding Claim 14: The rejection of claim 13 is incorporated; further Chen discloses changing the version indicator (col. 8, lines 59-63 'checksum is used ...

to determine whether an entity has been changed' and col. 11, lines 26-27 'the checksum ... is retrieved from the second C program database'). While not explicitly stated, the checksum does identify the version of the entity in that it distinguishes between two different versions of said entity.

Regarding Claim 15: Chen discloses an article of manufacture comprising a program storage device (col. 6, lines 50-52 'the external storage device) readable by a computer and tangibly embodying one or more programs of instructions (col. 6, lines 50-52 'the external storage device ... may be used for the storage of ... computer program code') executable by the computer to perform method steps for collecting persistent code coverage data for a computer program (col. 1, lines 11-14 'selective regression'), the computer program comprising program source code statements (col. 5, lines 33-52), the method comprising the steps of: identifying the computer program for which the persistent code coverage data should be collected (col. 2, lines 47-50 'a software system'); dividing the program source code statements of said computer program into a plurality of code coverage tasks (col. 2 lines 47-50 'partitioned into basic code entities'), generating a persistent unique name for each of the code coverage tasks (col. 9, lines 1-3 'all entities have the attributes ... kind, ... name' and col. 11, lines 20-21 'two entities match if they have the same name and entity kind') of said plurality of code coverage tasks; inserting coverage points into the computer program source code for each of the code coverage tasks to produce an instrumented program (col. 7, lines 7-9 'adding instrumentation'), compiling and linking the instrumented program into a program executable (col. 7, lines 9-11 'compiled by

a C compiler'), identifying a set of test cases from a plurality of test cases to be run for the code coverage data collection purposes (col. 11, lines 66-67 'determine which test units ... need to be re-run'); creating a code coverage database using the code coverage tasks and the identified set of test cases (col. 9, lines 32-35 'generate an entity trace list for each test unit'); running the program executable with a test case from the identified set of test cases (col. 7, lines 30-33 'for each of the N test units') and writing the information about the test case and the coverage points that are executed into an output file (col. 7, lines 40-44 'generates a function trace list'), until all the test cases have been run (col. 7, line 32 'for each of the N test units'); and processing the information contained in the output file into code coverage data and populating the code coverage database with said code coverage data (col. 9, lines 32-35 'the function trace lists ... are then used to generate an entity trace list').

Regarding Claim 20: The rejection of claim 15 is incorporated; further, Chen discloses modifying the computer program to produce a modified version of the computer program source code (col. 10. line 50-53 'after modifications have been made'); identifying a plurality of new, modified, and deleted code coverage tasks (col. 10, lines 63-66 'entity difference list') in said modified version of the computer program source code; generating a persistent unique name (col. 9, lines 1-3 'all entities have the attributes ... kind, ... name' and col. 11, lines 20-21 'two entities match if they have the same name and entity kind') for each of the new and modified code coverage tasks of said plurality of new, modified and deleted code coverage tasks; inserting coverage points (col. 7, lines 7-9 'adding

instrumentation') into the modified version of the computer program source code for each of the new and modified code coverage tasks to produce an instrumented modified version of the computer program source code; compiling and linking the instrumented modified version of the computer program source code (col. 7, lines 9-11 'compiled by a C compiler') into a modified program executable, identifying a new set of test cases (col. 11, lines 66-67 'the entity difference list is used ... to determine which tests units ... need to be re-run') from a plurality of test cases to be run for the code coverage data collection purposes on the new and modified code coverage tasks; altering the code coverage database to accommodate new, modified and deleted code coverage tasks (col. 12, lines 41-47 'new entity trace lists must be generated for each test unit') and the new set of test cases, and clearing any code coverage data for the modified code coverage tasks (col. 12, lines 41-43 'new entity traces list must be generated') from said code coverage database; running the modified program executable with a test case from the identified new set of test cases (col. 11, lines 66-67 're-run') and collecting code coverage data for the new and modified code coverage tasks (col. 7, lines 40-44 'generates a function trace list'), until all the test cases have been run (col. 7, line 32 'for each of the N test units'); and updating the code coverage database with the collected code coverage data (col. 9, lines 32-35 'the function trace lists ... are then used to generate') for the new and modified code coverage tasks; whereby the previously collected code coverage data for the non-affected code coverage tasks is preserved (col. 12, liens 41-47 'new entity trace lists must be generated ... covered by each of the

selected test units') from a previous version of the computer program to the modified version of said computer program eliminating the need for running the entire test bucket (col. 11, lines 66-67 'which test units ... need to be re-run').

Regarding Claim 21: The rejection of claim 20 is incorporated; further Chen discloses changing the version indicator (col. 8, lines 59-63 'checksum is used ... to determine whether an entity has been changed' and col. 11, lines 26-27 'the checksum ... is retrieved from the second C program database'). While not explicitly stated, the checksum does identify the version of the entity in that it distinguishes between two different versions of said entity.

Claims 2-3, 9-10, 16-17 are rejected under 35 U.S.C. 103(a) as being unpatentable over USPN 5,573,387 to Chen et al. (Chen) in view of 'Managing data through naming standards' by Winder, Software, IEEE, Volume: 7, Issue: 4, July 1990 (Winder).

Regarding Claims 2, 9, and 16: The rejections of claim 1, 8 and 15 are incorporated respectively; further Chen does not disclose using naming conventions. But does disclose attributes of each entity provide a unique identifier for said entity (col. 11, lines 20-21 'two entities match if they have the same name and entity kind.').

Winder teaches using a naming convention (pg. 85, col. 1, par. 3 'A naming standard fights ... ambiguity') in an analogous art for the purpose of managing data and eliminating ambiguity (pg. 85, col. 1, par. 4 'eliminating the ambiguity').

It would have been obvious to a person of ordinary skill in the art at the time of the invention to include a naming convention as taught by Winder in the naming of coverage tasks ('entities') as disclosed in Chen.

The modification would have been obvious because one of ordinary skill in the art would have been motivated to provide names for entities that would enable one to access the entities (Winder pg. 84, col. 3, par. 1-2 'determines your ability to access the information'), in Chen's disclosed system.

Regarding Claim 3, 10 and 17: the rejection of claims 2, 9 and 16 is incorporated; further, Chen does not disclose the naming convention comprising a module name, a version and a unique task identifier But does disclose maintaining similar attributes (i.e. col. 9, lines 1-3 'kind, file, name and checksum') where checksum is used to determine a function version (col. 8, lines 59-62 'determine whether an entity has been changed')

Winder teaches the use of a three part naming convention (pg. 85, col. 2, par. 1 'these data-element names are called primary, class, and modifier').

It would have been obvious to a person of ordinary skill in the art at the time of the invention to use Winder's three part naming convention (pg. 85, col. 2, par. 1) populated with the data gathered in Chen col. (col. 9, lines 1-3) to label the entities disclosed in Chen (col. 8, lines 1-4) thereby creating a unique naming convention comprising a computer program module name (col. 8, lines 44-45 'file'), a version indicator (col. 8, lines 51-52 'checksum'), and a unique code coverage task identifier (col. 8, lines 46-47 'name').

The modification would have been obvious because one of ordinary skill in the art would have been motivated to provide names for entities that would enable one to access the entities (Winder pg. 84, col. 3, par. 1-2 'determines your ability to access the information'), in Chen's disclosed system.

Claims 4, 11, and 18 are rejected under 35 U.S.C. 103(a) as being unpatentable over USPN 5,573,387 to Chen et al. (Chen) in view of USPN 5,778,169 to Reinhardt (Reinhardt).

Regarding Claims 4, 11, and 18: The rejections of claims 1, 8, and 15 are incorporated, respectively; further, Chen does not disclose that the code coverage database comprises a table, the table comprising a row for each test case in said identified set of test cases and a column for each code coverage task of said plurality of code coverage tasks, said column comprising an indicator at each row indicating coverage status for said code coverage task. But does foresee the need to allow a user to determine which test units would need to be re-run if a hypothetical change were made to the software system (col. 12, lines 55-58).

Reinhardt teaches the code coverage database comprises a table (col. 6, lines 16-18 'test coverage matrix'), the table comprising a row for each test case (col. 6, lines 24-25 'names of the tests') in said identified set of test cases and a column for each code coverage task (col. 6, lines 25-26 'coverage point names') of said plurality of code coverage tasks, said column comprising an indicator at each row indicating coverage status for said code coverage task (col. 6, lines 26-

27 'the relationship between the tests and coverage points'), in an analogous art for the purpose of providing programmers with knowledge of which coverage points are executed by which tests (col. 6, lines 34-35 'view the test coverage matrix')

It would have been obvious to a person of ordinary skill in the art at the time of the invention to use the matrix taught by Reinhardt in the regression test system of Chen to display the program's code coverage data to a programmer (Chen col. 12, lines 55-58).

The modification would have been obvious because one of ordinary skill in the art would have been motivated to allow programmers to easily identify regression tests that test possible source code changes (Reinhardt, col. 2, lines 62-63).

Claims 5, 12 and 19 are rejected under 35 U.S.C. 103(a) as being unpatentable over USPN 5,573,387 to Chen et al. (Chen).

Regarding Claims 5, 12, and 19: The rejections of claims 1, 8, and 15 are incorporated respectively; further, Chen does not disclose that the computer program comprises program source code statements written in a hardware description language. But does teach that his invention may be 'applied to the selective regression testing of software systems written in other languages' (col. 9, lines 22-24).

It would have been obvious to one of ordinary skill in the art at the time of the invention to implement selective regression testing, as detailed by Chen, for a hardware description language.

The modification would have been obvious because one of ordinary skill in the art would have been motivated to provide a system to regression test software written in languages other than C (col. 9, lines 22-24).

(10) Response to Argument

Rejections 1, 3, and 4

"Persistent"

In the second paragraph on pg. 9 Appellant states

> First, the Examiner effectively has read out the word "persistent" from the claims. Appellants have used that term to define code coverage tasks that persist across software versions.

Examiner respectfully disagrees. The concept of persistence is fundamentally necessary to the nature of Chen's invention (i.e. col. 1, lines 12-14 'a system ... that identifies which subset of a test suite must be run in order to test a new version of a software system'). If the data gathered for an old version did not persist at least until the new version had been written it would be impossible to compare the two.

Additionally, it can be seen from col. 8 that the 'name attribute' is being stored in a 'C program database', thus explicitly disclosing persistence. Further in Chen's disclosed system, the entities in different software versions are compared by checking if names match or not. See col. 15, lines 45-48 and col. 11, lines 20-21. The names could not be compared if their names did not persist across versions.

"Unique Name"

Chen discloses generating identifying data for each of his "entities" (col. 9, lines 1-3 'all entities have the attributes ... kind, ... name') and discloses comparing these "entities" across software version based on this identifying data (col. 11, lines 20-21 'two entities match if they have the same name and entity kind'). Claim 1 does not recite a limitation requiring that the "persistent unique name" be stored in memory as a continuous string, consequently any combination of Chen's "attributes", which are used to identify an "entity" can be read as "generating a ... name".

Further, examiner agrees with Appellant, "having a unique name for a function would be an indispensable part of writing source code" (see the paragraph bridging pg. 13-14 of Appellant's brief). Chen's "name" attribute is taken from the function name (col. 8, lines 46-47 'The name attribute indicates the name of the function') and consequently must also be unique.

Still further, it can be seen from col. 11, lines 20-21 ('two entities match if they have the same name and entity kind'), among other places, that Chen's invention (col. 1, lines 11-13 'method for selective regression testing') could not function properly if there was not a unique identifier for each entity. For example false matches would occur, making it impossible to properly determine which entities have been added deleted or changed (col. 10, line 63-col.11, line 4).

"Code Coverage Task"

Starting in the last full paragraph on pg. 11, Appellant states:

> From the forgoing, it can be appreciated that, while the Examiner has attempted to read the claimed "code coverage tasks" on Chen's "basic code entities," the two things are quite different from each other. ... Actually, the

> claimed "code coverage tasks" correspond to a Chen program, or to a block of a Chen program, and not to a function, variable, type or macro.

Examiner respectfully disagrees.

Appellant's specification defines a "code coverage task" as:

> A code coverage task is a basic block of code for which an execution of a test returns a true value if the testing requirement of the task is fulfilled and a false value if the testing requirement of the task is not fulfilled.

Chen discloses executing a test (col. 9, lines 32-34 'generate an entity trace list'), which returns a true value for each "entity" which has been executed (inclusion in the list) and a false value for those, which have not (exclusion from the list). Thus it is clear that Chen's "entities" are within the scope defined by applicant.

Further, the fact that Chen discloses additional functionality (i.e. determining code coverage related to variables, types and macros) does not represent a patentable distinction between Chen and the instant application.

Starting in the paragraph bridging pp. 11-12 Appellant states:

> However, the Examiner's expansive reading of this language ignores the following sentence from that quote which gives an example of the kind of "other alternative ways" that the inventors contemplated: "For example, coverage tasks could be at module level, block level or statement level and could be identified manually rather than automatically and could be based on the user's needs".
> Thus, the claimed code coverage tasks are very different from Chen's "entities".

Examiner respectfully disagrees. The definition given in Appellants specification (pg. 12-13) is very broad and discloses that 'coverage tasks could be at module level, block level or statement level ... and could be based on the user's needs' and that 'Those of ordinary skill in the art will recognize that there are other alternative ways to divide a program source code into coverage tasks', and does

not restrict a "code coverage task" to the examples given. Chen's discloses function correspond to a module level code coverage task and fall well within the scope of a "code coverage task" as defined by Appellant.

Appellant goes on to state:

> There is a tremendous difference between giving a persistent unique name to a code block, for purposes of having it left out of test suite running, and naming a function.

Respectfully, Examiner points to Chen's col. 7, line 64- col. 8, line 4 where he discloses, "The C information abstractor 175 generates the C program database 177 [containing] the entities that comprise the system ... and attributes of the entities". In this passage, Chen's "C information abstractor" generates, among other things, the "name attribute" and is clearly doing more than simply "naming a function".

In the second full paragraph on pg. 12, Appellant states:

> Based on the foregoing, when the present invention inserts coverage points into the computer program source code for each of the code coverage tasks, as claimed for example in claim 1, the portion of Chen on which the Examiner relies at col. 7, lines 7-9 referring to the adding of instrumentation, has nothing to do even with the Chen "entities" which the Examiner is attempting to the claimed code coverage tasks.

Respectfully, the relevant part claim 1 recites:

> Inserting coverage points into the computer program source code for each of the code coverage tasks to produce an instrumented program

Examiner respectfully notes that, as can be seen in col. 9, lines 32-57, 'function trace lists 161 and 163 through 165' generated by the instrumented code contain sufficient data to determine which 'entities' or 'code coverage tasks' have been

executed and consequently there must be instrumentation 'for' each 'entity', thereby meeting the limitations of the claim.

Rejection 2

In the paragraph bridging pp. 13-14 Appellant states:

> Therefore, concerning claims 2, 9, and 16, since Chen's entities do not correspond to the claimed code coverage tasks, the unique naming of these claims finds no response in Chen, as the Examiner acknowledges, but also finds no response in Winder because there is simply no motivation to make any kind of suggestions (assuming for the sake of argument that there are suggestions) in Winder to modify the teachings of Chen

Chen recognizes the usefulness of the data items recited in claim 3 (col. 9, lines 1-3 'kind, file, name and checksum'). Winder indicates that combination of various data items into a single string (i.e. use of a naming convention) was well known at the time of the invention (pg. 85, col. 2, par. 1 'these data-element names') and further teaches such a combination results in a improved means of retrieving such data (pg. 85, col. 1 'a naming standard fights the major obstacle to information management: ambiguity'). Such disclosures provide ample motivation to one of ordinary skill in the art to combine the various attributes gathered by Chen into a string attribute, thereby allowing a user to retrieve all the information needed in a single data item, using winder's teaching.

Further it should be pointed out that Examiner did not acknowledge 'the unique naming of these claims finds no response in Chen'. What was acknowledged was that Chen does not explicitly disclose use of a 'naming convention'. A naming convention and a unique name are not the same thing. As pointed out in

the final rejection and addressed on pg. 17 or the Examiners Answer, Chen clearly teaches a "unique name".

(11) Related Proceeding(s) Appendix

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

(12) Evidence Appendix

Appellant has not entered or relied upon any evidence in this appeal.

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

Jason Mitchell

1/31/06

Conferees:

Kakali Chaki

KAKALI CHAKI
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100

Tuan Dam

TUAN DAM
SUPERVISORY PATENT EXAMINER